perfSONAR for testing network configuration parameters

Tim Chown (Jisc)

4th European perfSONAR workshop, 16 May 2024

Agenda

A slightly different use case for perfSONAR

- Normally perfSONAR is used for day-to-day network monitoring
- But it can also help us evaluate the effect of different network tuning parameters and to try emerging options such as TCP-BBR

- Topics…
  - The importance of network tuning
  - Leveraging *pscheduler* as a test harness
  - Jisc perfSONAR nodes
  - Some examples
  - Thoughts – what might we do to better support this use case?

The importance of network tuning

Why tune?

What's the rationale?

- Default operating system network tuning configurations are set for common application use cases
- Most use cases don't need any special configuration

- In R&E networks, we often have quite demanding applications, particularly around data-intensive science and very large data transfers
  - e.g., 10TB in an hour needs about 20Gbps
    or 1PB in 1 day needs around 100Gbps

- For happy researchers we need to optimise our use of available capacity

What might we tune?

What properties might we change to optimise throughput?

- Examples:
  - TCP Congestion control algorithm (CCA)
  - Maximum Transmission Unit (MTU) (packet size)
  - TCP window size (buffers)
  - Number of streams used by an application
  - Pacing (maximum throughput per stream)
  - Queueing algorithm
  - …

- There's a lot of great information on tuning at https://fasterdata.es.net/
  for various host OSes and network devices

Congestion control example - BBRv3

Bottleneck Bandwidth and Round-trip (BBR)

- BBRv3 was announced at the July 2023 IETF meeting
  - See https://datatracker.ietf.org/meeting/117/materials/slides-117-ccwg-bbrv3-algorithm-bug-fixes-and-public-internet-deployment-00 (an update from Google, who created the first versions and use it in production)
- Key innovation is not relying on packet loss to detect congestion
  - BBR measures throughput and latency to determine its send rate
- Open source, available on github:
  - https://github.com/google/bbr/blob/v3/README.md
- Holds a lot of potential benefit for large-scale R&E data transfers
  - Resilient to loss, adapts well to long distance scenarios (long fat pipes!)
- Not (yet) in production Linux distributions, but it should be evaluated

MTU - jumbo frames

Using large MTU packets

- The default MTU for ethernet LANs is 1500
- Using 9000 MTU can give improved throughput
  - Faster TCP ramp-up, fewer packets per second to process
  - See https://network.switch.ch/pub/tools/tcp-throughput/
- There was a WLCG proposal in 2018 to use jumbo frames:
  - https://indico.cern.ch/event/725706/contributions/3120030/attachments/1743507/2821722/LHCONE-MTU-recommendation.pdf
- Many WLCG sites do now run 9000 MTU, but many do not
- It would be very useful to measure the performance benefit of 9000 vs 1500 MTU, or other MTU values

TCP window size

How much data can be in flight?

- When TCP is used, its sending rate is limited by how much data can be in flight, before acknowledgements are received
- The “Bandwidth-delay product”
- For greater capacity paths, and higher round-trip time paths, the BDP will be greater
- Again, see https://network.switch.ch/pub/tools/tcp-throughput

- It’s useful to be able to test (Linux) window settings
- Though modern Linuxes do have some auto-tune support
- As you might guess, FasterData is a good source of information

Number of parallel streams

An application-oriented parameter

- If an application uses one TCP stream, its throughput will be determined by the performance of that stream
- By running multiple concurrent streams, the impact of one stream experiencing poor throughput (typically through packet loss with traditional TCP) is reduced
- Globus and its GridFTP tool use 4 streams by default
- It’s useful to be able to test the effect on overall throughput from running 1, 2, 4, or more concurrent streams

Jisc

Packet pacing

Controlling traffic sending rates

- A relatively new topic in the WLCG's Research Networking Technical Working Group (RNTWG)
  - Held a kick off meeting in May 2023
  - https://indico.cern.ch/event/1287745/sessions/494231/#20230525
- Pacing has potential to reduce traffic microbursts to improve overall throughput
- Particularly useful where shallow buffer devices are in use
- In principle there should be less buffering and thus fewer drops
- "Competing" streams can also be fairer to each other
- This can be tested by rate-limiting TCP streams

- Note that BBRv3 effectively paces itself, so will be interesting to compare

Leveraging *pscheduler* as a test harness

How can we use pscheduler?

Running comparisons

- You can of course run tests from your own perfSONAR server
- But a key feature of *pscheduler* is that it can also **run third party tests between two remote perfSONAR measurement points**
  - Requires those servers to be open for remote tests
- Tests can be run on demand or scheduled via pSConfig or other means
- If we want to make comparisons we could either
  - Re-tune the perfSONAR server and continue to run tests as normal, giving “before and after” measurements, perhaps for a day or a week
  - Use *pscheduler* to vary the server tuning on a per-test basis, which may give a better comparison given the network load is more likely to be similar
- Then the results and differences can be viewed or analysed

What parameters does *pscheduler* support?

Examples for throughput tests

- CCA: --congestion
- MTU: --mss (actually the TCP maximum segment size)
- TCP window size: --window-size
- Number of streams: --parallel
- Pacing: --bandwidth

- You can use any combination of the above
- See https://docs.perfsonar.net/pscheduler_ref_tests_tools.html

What else might we study?

Examples

- IPv6
  - Relative performance to IPv4
  - PMTUD operation, flow label transparency, …
- MTU limitations
  - Are there MTU misconfigurations?
  - We could detect MTU changes on a path using a *tracepath* test
- Explicit Congestion Notification (ECN)
  - Test with and without ECN enabled; is throughput improved?
  - BBR can leverage ECN

Jisc perfSONAR nodes

Jisc and network performance

Some Jisc resources

- Janet network test facilities
  - https://www.jisc.ac.uk/guides/using-the-janet-network-performance-test-facilities
  - Includes 10G and 100G perfSONAR servers
  - Raul has been testing 5.1 beta, and has some nice Grafana views!

- Research Network Engineering community:
  - https://www.jisc.ac.uk/get-involved/research-network-engineering-rne-community-group
  - Includes a BBR presentation in the archive
  - Next call is on perfSONAR 5.1
  - JiscMail list – *rne@jiscmail.ac.uk* - join at https://www.jiscmail.ac.uk/RNE

Some examples

Configuring servers for tests

Must ensure pscheduler can use a full range of parameters

- Server set open for 3rd party testing
- BBRv3 installed (not necessarily as the default CCA)
- 9000 MTU enabled
- IPv6 enabled
- Enhanced window/buffer size settings by default (e.g., using settings from FasterData)

Example: cubic vs BBR

CUBIC

```
$ pscheduler task throughput  --source ps-london-bw.perf.ja.net --dest a.n.other.perfsonar --congestion cubic

* Stream ID 5

Interval       Throughput     Retransmits    Current Window
0.0 - 1.0      5.10 Gbps      3852           13.24 MBytes
1.0 - 2.0      5.51 Gbps      0              13.66 MBytes
2.0 - 3.0      5.66 Gbps      0              14.03 MBytes
3.0 - 4.0      5.82 Gbps      0              14.35 MBytes
4.0 - 5.0      5.89 Gbps      0              14.63 MBytes
5.0 - 6.0      6.01 Gbps      0              14.86 MBytes
6.0 - 7.0      6.12 Gbps      0              15.06 MBytes
7.0 - 8.0      6.16 Gbps      0              15.22 MBytes
8.0 - 9.0      6.20 Gbps      0              15.35 MBytes
9.0 - 10.0     6.30 Gbps      0              15.45 MBytes

Summary
Interval       Throughput     Retransmits    Receiver Throughput
0.0 - 10.0     5.88 Gbps      3852           5.80 Gbps
```

Example: cubic vs BBR

BBR

```
$ pscheduler task throughput  --source ps-london-bw.perf.ja.net --dest a.n.other.perfsonar --congestion bbr

* Stream ID 5

Interval      Throughput     Retransmits    Current Window
0.0 - 1.0     13.51 Gbps     113552         44.39 MBytes
1.0 - 2.0     18.54 Gbps     40645          113.13 MBytes
2.0 - 3.0     18.65 Gbps     37872          116.69 MBytes
3.0 - 4.0     19.29 Gbps     15092          104.01 MBytes
4.0 - 5.0     18.17 Gbps     8263           102.90 MBytes
5.0 - 6.0     15.82 Gbps     5605           103.43 MBytes
6.0 - 7.0     17.86 Gbps     52285          110.02 MBytes
7.0 - 8.0     19.10 Gbps     31216          112.44 MBytes
8.0 - 9.0     16.70 Gbps     73078          98.93 MBytes
9.0 - 10.0    18.71 Gbps     77432          114.61 MBytes

Summary
Interval      Throughput     Retransmits    Receiver Throughput
0.0 - 10.0    17.63 Gbps     455040         17.40 Gbps
```

Note: the retransmissions
are **much** greater with BBR

The destination has been
anonymised given there is
some packet loss

Also note that BBR is only
Required sender side, but other
Network configuration will matter

Jisc

Example: cubic vs BBR (CERN – Janet)

CUBIC

```
$ pscheduler task throughput  --source pse01-gva.cern.ch --dest ps-london-bw.perf.ja.net --congestion cubic

* Stream ID 5
Interval       Throughput     Retransmits    Current Window
0.0 - 1.0      451.29 Mbps    0              2.85 MBytes
1.0 - 2.0      5.89 Gbps      503            17.73 MBytes
2.0 - 3.0      8.79 Gbps      0              18.64 MBytes
3.0 - 4.0      8.67 Gbps      0              19.47 MBytes
4.0 - 5.0      9.46 Gbps      0              20.23 MBytes
5.0 - 6.0      9.37 Gbps      0              20.91 MBytes
6.0 - 7.0      9.46 Gbps      0              21.53 MBytes
7.0 - 8.0      9.81 Gbps      0              21.92 MBytes
8.0 - 9.0      10.41 Gbps     0              22.57 MBytes
9.0 - 10.0     9.96 Gbps      0              23.02 MBytes

Summary
Interval       Throughput     Retransmits    Receiver Throughput
0.0 - 10.0     8.23 Gbps      503            8.21 Gbps
```

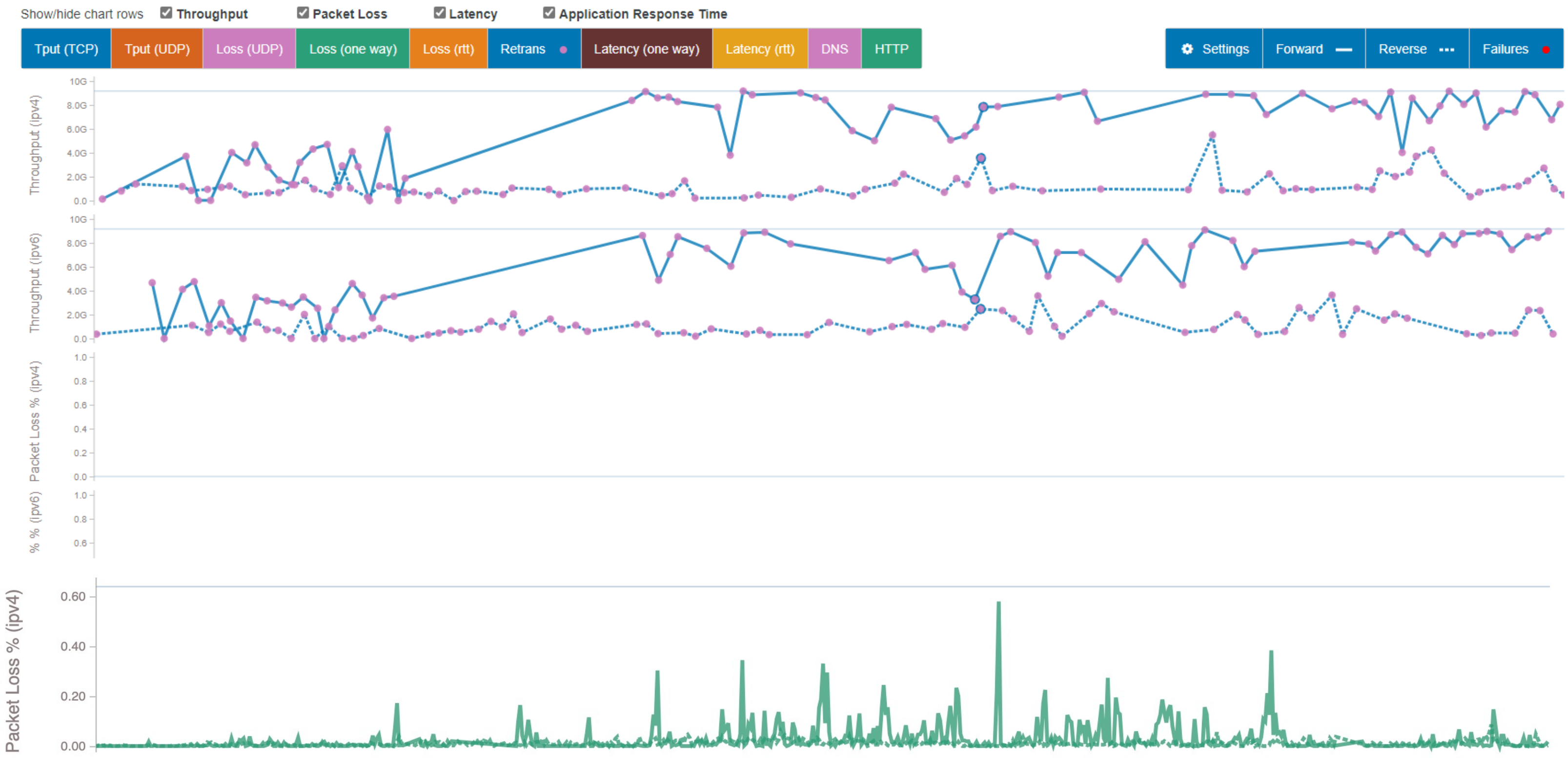
Example: cubic vs BBR (CERN – Janet)

BBR

```
$ pscheduler task throughput  --source pse01-gva.cern.ch --dest ps-london-bw.perf.ja.net --congestion bbr

* Stream ID 5

Interval        Throughput      Retransmits     Current Window
0.0 - 1.0       9.04 Gbps       26866           74.03 MBytes
1.0 - 2.0       12.81 Gbps      0               59.20 MBytes
2.0 - 3.0       10.84 Gbps      349             58.23 MBytes
3.0 - 4.0       11.73 Gbps      19408           63.83 MBytes
4.0 - 5.0       12.73 Gbps      1140            54.63 MBytes
5.0 - 6.0       10.35 Gbps      10339           52.01 MBytes
6.0 - 7.0       12.55 Gbps      0               49.73 MBytes
7.0 - 8.0       12.46 Gbps      0               49.56 MBytes
8.0 - 9.0       12.58 Gbps      0               50.88 MBytes
9.0 - 10.0      12.49 Gbps      0               49.69 MBytes

Summary
Interval        Throughput      Retransmits     Receiver Throughput
0.0 - 10.0      11.76 Gbps      58102           11.74 Gbps
```

In this case the path from CERN to Janet is a good, relatively short and loss-free one, so there is less benefit from BBR in terms of the throughput achieved

US-UK example: BBR improvement (lossy destination)



Thoughts

What might we do?

What could we add to perfSONAR to support this use case?

- perfSONAR commonly used to measure network characteristics over time
- The capability for perfSONAR to test network configuration changes, and new tools such as BBR, is often overlooked
- WLCG Data Challenge 24 illustrated the power of *pscheduler*
- Need to ensure open nodes are available
- **How might we best schedule comparative tests?**
- **How can we make useful visualisations for comparitive measurements?**
- A lot of potential here!

Tim Chown

tim.chown@jisc.ac.uk

netperf@jisc.ac.uk

Tel 01325 822106

customerservices@jisc.ac.uk

jisc.ac.uk